Storage

How to store data on the client

Sync / Async

Synchronous APIs

Blocks the thread

Easy to use

Just ordinary functions!

Asynchronous APIs

Non blocking

Uses callbacks or promises

Adds complexity

For everything that can take “time”

Callback Pattern APIs

```
myCallbackAPIv1.get('resource', (error, result) => {
  if (error) {
    // Handle error
  }
  // Work with result
})
```

```
let connection = myCallbackAPIv2.get('resource')

connection.addEventListener('error', (event) => {
  // Handle errors
})

connection.addEventListener('success', (event) => {
  // Handle result
})
```

Promisified APIs

```
myPromiseAPI.get('resource').then( (result) => {
  // Work with the result
}).catch(error => {
  // Handle the error
})
```

```
try {
  let result = await myPromiseAPI.get('resource')
} catch (error) {
  // Handle the error
}
```

in an async context

Lagen om elektronisk kommunikation

> *Uppgifter får lagras i eller hämtas från en abonnents eller användares terminalutrustning endast om abonnenten eller användaren får tillgång till information om ändamålet med behandlingen och samtycker till den. Detta hindrar inte sådan lagring eller åtkomst som behövs för att överföra ett elektroniskt meddelande via ett elektroniskt kommunikationsnät eller som är nödvändig för att tillhandahålla en tjänst som användaren eller abonnenten uttryckligen har begärt.*
>
> */Lagen om elektronisk kommunikation Kap 6, 18*

“Cookielagen”

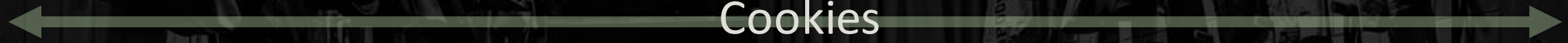
How to store data?

Persistent*

Non-persistent

Non-volatile

Volatile



Cookies

Webstorage –
Local Storage

Webstorage –
Session Storage

IndexDB

Cache API

File System Access API

** Persistant data in the user browser is always some what volatile since the user at any time can switch browser or clear the stored data.*

Same Origin

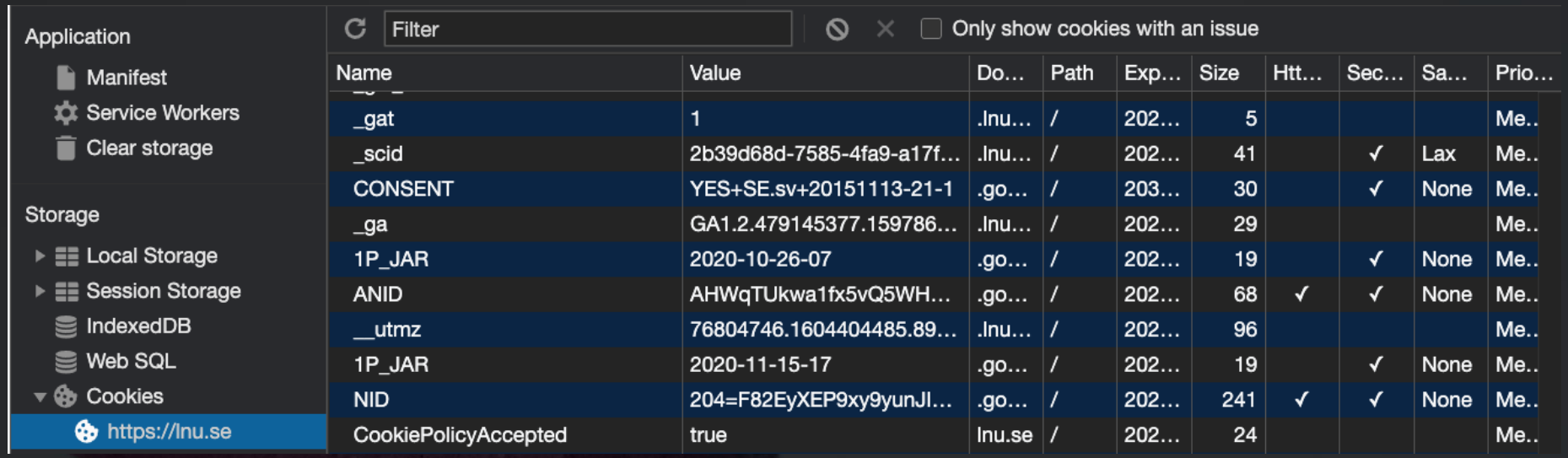
`https://gitlab.lnu.se:443`

`scheme (protocol)` — `host name` — `port`

Origin: `scheme + host name + port`

Site: `~ host name`

Cookies



Name	Value	Do…	Path	Exp…	Size	Htt…	Sec…	Sa…	Prio…
_gat	1	.lnu…	/	202…	5				Me…
_scid	2b39d68d-7585-4fa9-a17f…	.lnu…	/	202…	41		✓	Lax	Me…
CONSENT	YES+SE.sv+20151113-21-1	.go…	/	203…	30		✓	None	Me…
_ga	GA1.2.479145377.159786…	.lnu…	/	202…	29				Me…
1P_JAR	2020-10-26-07	.go…	/	202…	19		✓	None	Me…
ANID	AHWqTUkwa1fx5vQ5WH…	.go…	/	202…	68	✓	✓	None	Me…
__utmz	76804746.1604404485.89…	.lnu…	/	202…	96				Me…
1P_JAR	2020-11-15-17	.go…	/	202…	19		✓	None	Me…
NID	204=F82EyXEP9xy9yunJI…	.go…	/	202…	241	✓	✓	None	Me…
CookiePolicyAccepted	true	lnu.se	/	202…	24				Me…

```
console.log(document.cookie) // "gID=132446fdgv24rsdf233rdsf3f32r1d"

document.cookie = "name=Johan"

console.log(document.cookie) // "gID=132446fdgv24rsdf233rdsf3f32r1d; name=Johan"
```

- Often used to keep a session against the server
- Use max 4096 bytes per cookie (RFC6265)
- No more than 50 cookies per domain. (RFC6265)

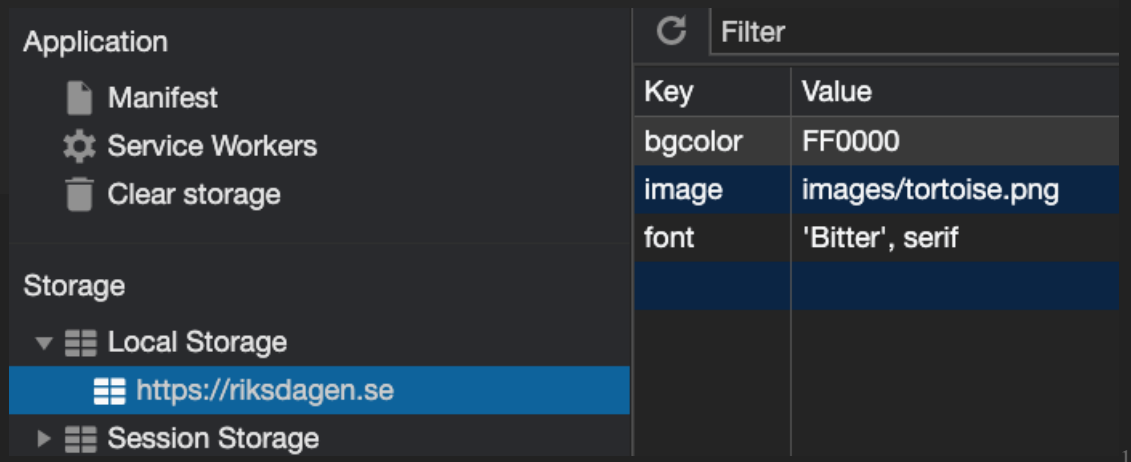
Web Storage

Synchronous API ⚠
2 -10 MB
Not accessible via Workers

setItem, getItem, removeItem

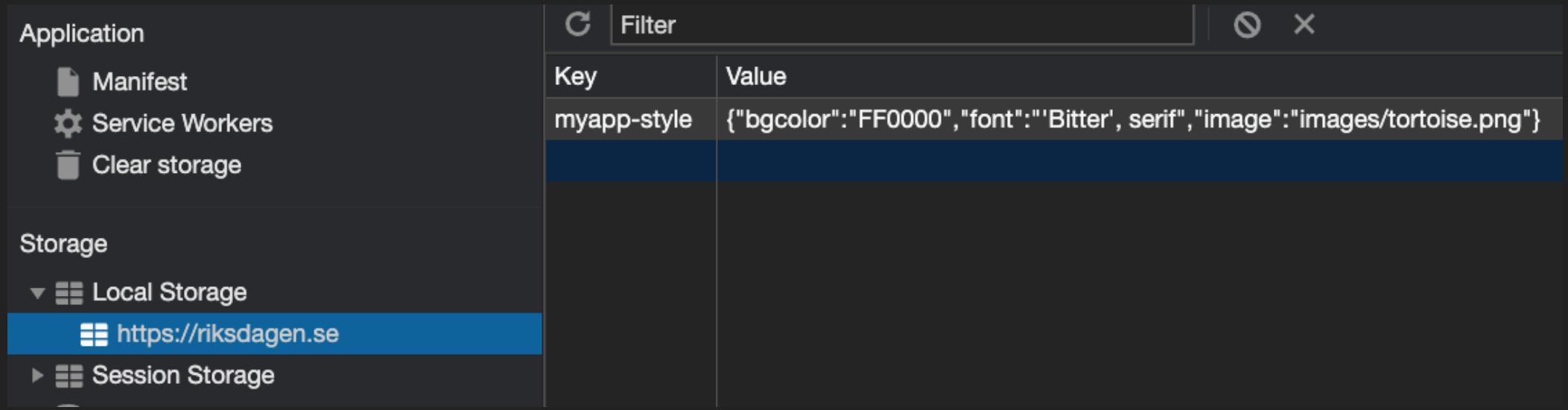
```
function populateStorage() {
  window.localStorage.setItem('bgcolor', document.getElementById('bgcolor').value)
  window.localStorage.setItem('font', document.getElementById('font').value)
  window.localStorage.setItem('image', document.getElementById('image').value)

  setStyles()
}

function setStyles() {
  let currentColor = window.localStorage.getItem('bgcolor’)
  let currentFont = window.localStorage.getItem('font’)
  let currentImage = window.localStorage.getItem('image')
...
```

What happens if two apps on the same origin uses the key ‘image’?



JSON in storage

```
const style = {
  bgcolor: document.getElementById('bgcolor').value,
  font: document.getElementById('font').value,
  image: document.getElementById('image').value
}

window.localStorage.setItem('myapp-style', JSON.stringify(style))

...

style = JSON.parse(window.localStorage.getItem('myapp-style'))
```



What happens if the user opens two browser instances at the same time?

IndexDB

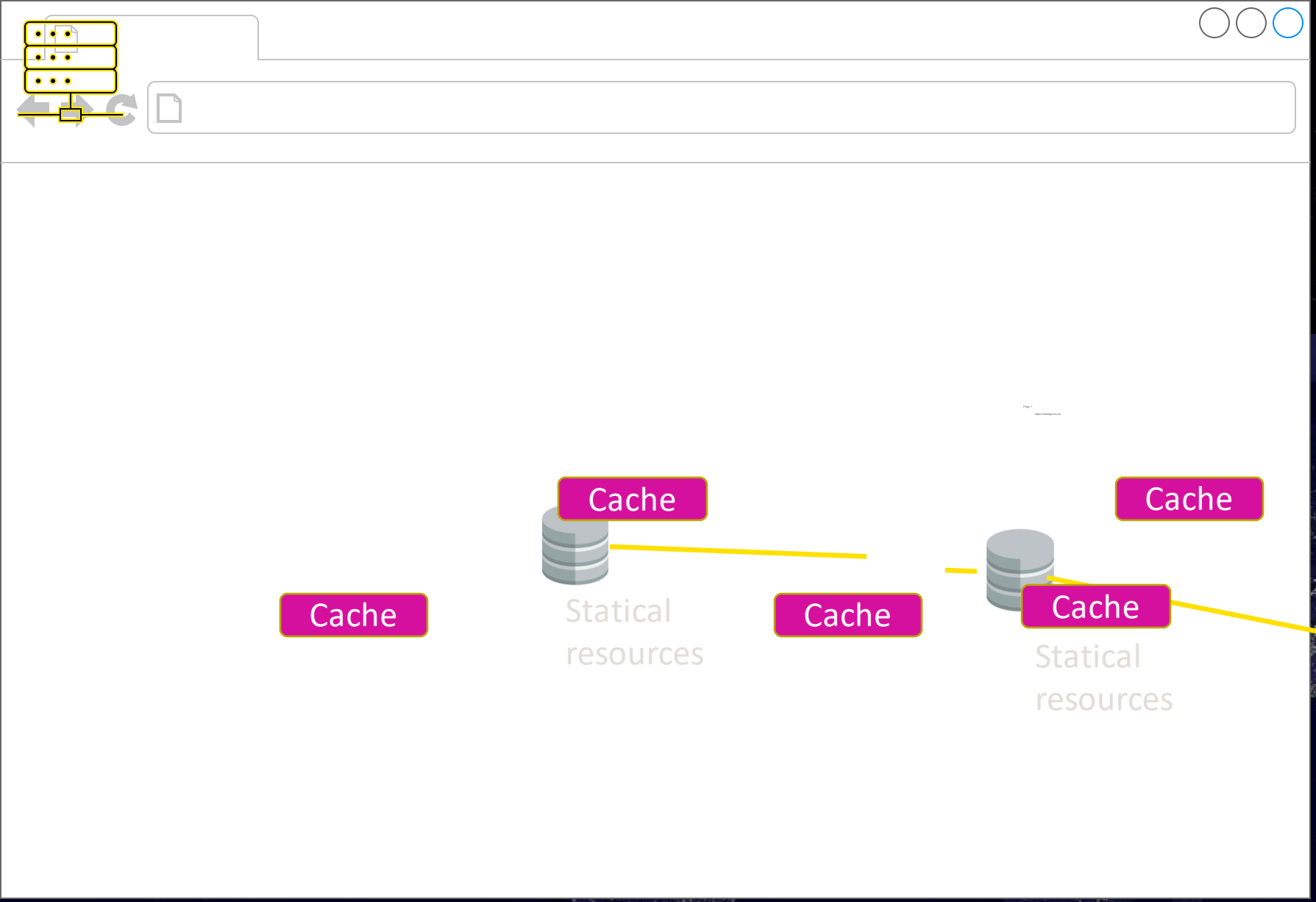
Asynchronous API (callbacks)
1GB – 60% of available disc space
Accessible via Workers

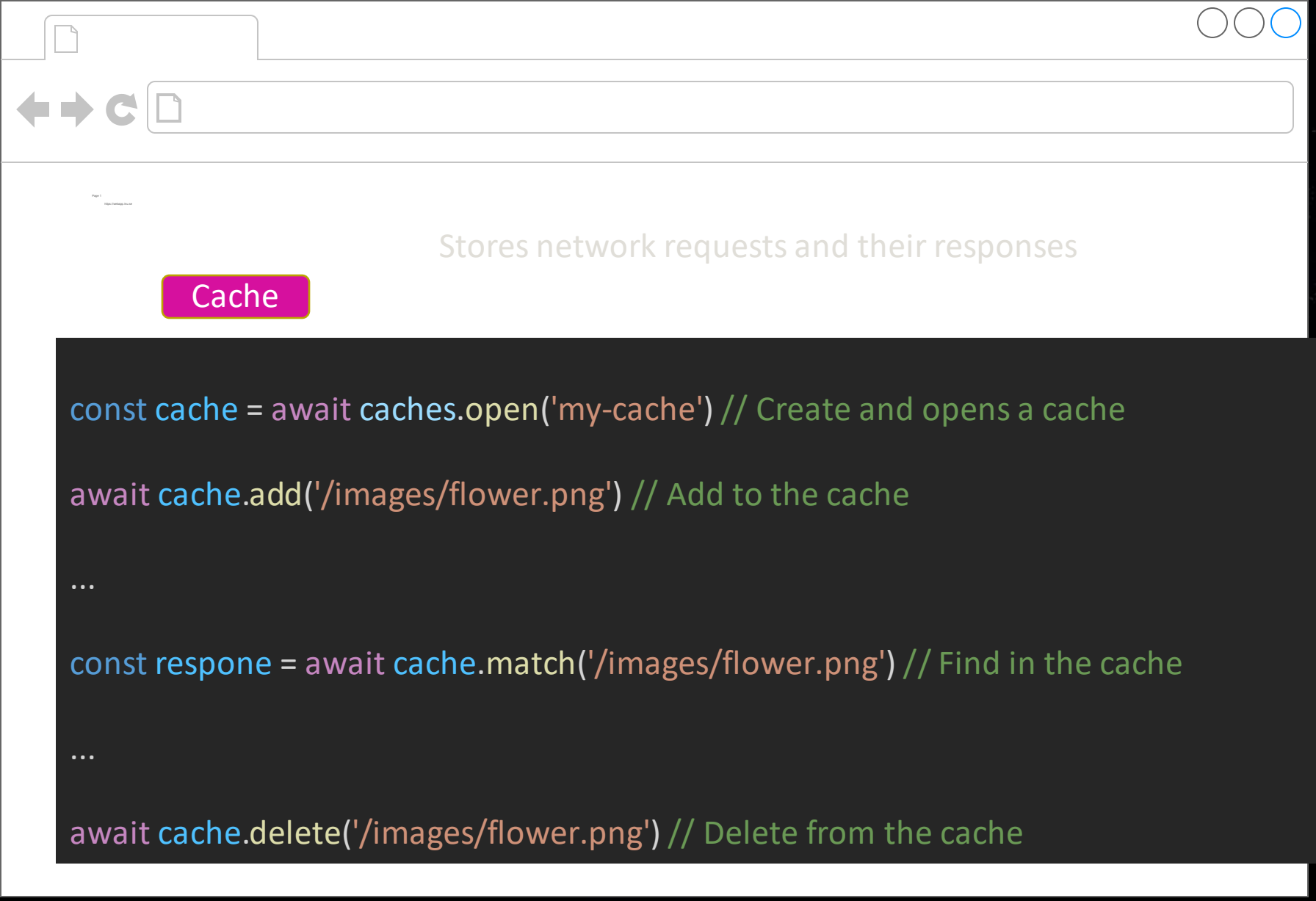
- IndexedDB databases store key-value pairs.
- IndexedDB is object-oriented.
- The IndexedDB API is mostly asynchronous.
- IndexedDB uses a lot of requests.
- IndexedDB does not use Structured Query Language (SQL).
- IndexedDB uses DOM events to notify you when results are available.
- IndexedDB adheres to a same-origin policy.
- IndexedDB is built on a transactional database model.

ACID

- Atomic
  - All or nothing
- Concistent
  - Follow the rules of the DB
- Isolated
  - Do not affect other transactions
- Durable
  - Written to permanent storage

https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Concepts_Behind_IndexedDB


Cache
Cache
Statical
resources
Cache
Cache
Cache
Statical
resources
Cache
Statical
resources

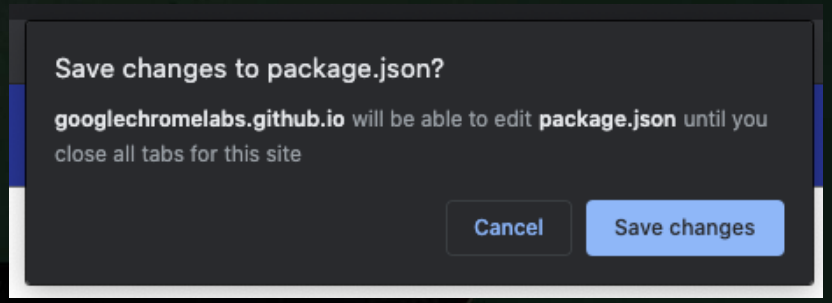


Stores network requests and their responses

Cache

```
const cache = await caches.open('my-cache') // Create and opens a cache

await cache.add('/images/flower.png') // Add to the cache

...

const respone = await cache.match('/images/flower.png') // Find in the cache

...

await cache.delete('/images/flower.png') // Delete from the cache
```

File System Access API




```
let fileHandle
butOpenFile.addEventListener('click', async () => {
  [fileHandle] = await window.showOpenFilePicker()
  const file = await fileHandle.getFile()
  const contents = await file.text()
  textArea.value = contents
})
```

Need a secure context and need to be called from within a user gesture.

https://web.dev/file-system-access/

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
	12-18		4-73		10-60											
6-10	1 79-85	2-81	1 74-85	3.1-13.1	1 62-71	3.2-13.7		2.1-4.4.4	12-12.1				4-11.2			
11	2 86	82	2 86	14	2 72	14	all	81	59	86	82	12.12	12.0	10.4	7.12	2.5
		83-84	2 87-89	TP												

Storage Manager API

Asynchronous API (promises)
Accessible via Workers

- ✓ DOMException “QuotaExceededError”

```
navigator.storage.estimate().then(est => { console.log(est.quota)})
```

```
navigator.storage.estimate().then(est => { console.log(est.quota/1073741824 + ‘GB’)})
```

```
navigator.storage.estimate().then(est => { console.log(est.usage + ‘ %’)})
```

Application Storage Section in the browser